CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

May 12, 2010

Optimization and Performance of HF PMT Hit Cleaning Algorithms Developed Using Collision Data at $\sqrt{s} = 0.9$ and 2.36 TeV

F. Chlebana

Fermi National Accelerator Laboratory, Batavia, IL, USA

I. Vodopiyanov

Florida Institute of Technology, Melbourne, FL, USA

V. Gavrilov

Institute for Theoretical and Experimental Physics, Moscow, Russia

D. Ferencek, F. Santanastasio, J. Temple

University of Maryland, College Park, MD, USA

Abstract

We present a study on the optimization and the performance of the noise cleaning algorithms developed to identify HF PMT hit candidates observed during the December 2009 data taking at $\sqrt{s} = 0.9$ and 2.36 TeV. Several Monte Carlo samples are used to evaluate the efficiency of these noise cleaning algorithms on real energy reconstructed in single HF channels, jets in HF and $\not\!\!E_{\mathrm{T}}$. We give an estimation of the expected rate of HF PMT hits at $\sqrt{s} = 7$ TeV. We also provide a CMSSW tool to implement user-defined noise cleaning algorithms, depending on the needs of each physics analysis. We finally propose a default HF PMT hit noise cleaning algorithm for the start-up at $\sqrt{s} = 7$ TeV.

1 Introduction

Commissioning studies performed during past test beams have identifed anomalous noise (*i.e.* not due to regular electronic pedestal noise) in the forward hadronic calorimeter (HF) []. The source of such anomalous signals are energetic charged particles, directly impinging upon the window of an HF Photo-Multiplier (PMT), generating Cherenkov light, and thereby producing an abnormally large apparent energy signal for a single HF channel (the one associated to that PMT). Due to the nature of the signal generation, the energy spectrum of such noise is realtively well defined, with a peak at $E \approx 100$ GeV and pronounced tails at higher energy values. Being in the very forward region, the transverse energy spectrum of such noise is constrained at relatively low E_T values, *i.e.* an energy of 100 GeV correponds to $E_T = E/\cosh(\eta)$ of ≈ 10 (1.3) GeV at $\eta = 3$ (5).

2 Handles to Identify HF PMT Hits

In this section we discuss possible ways to identify the HF PMT window hits. The main handles to identify the HF PMT hits are the pulse shape/timing and topology of the hits. Since the charged particles hitting the PMT window produce the Cherenkov light, the pulse shape of the PMT hits is expected to be contained within one 25-ns time sample just as the real signal created by particles depositing energy in HF. At the same time the PMT hit signal is expected to be earlier in time (by ~ 3–5 ns) due to the fact that the Cherenkov light produced in HF fibers needs additional time to reach the PMT's (quartz fibers have a high index of refraction, $n = 1.458$). In order to employ this timing difference to identify the PMT hits, a very precise timing phase alignment in HF is needed. At the time of writing this note the phase alignment has not been performed yet so the pulse timing cannot be used to identify the HF PMT hits.

The second handle to identify the HF PMT hits is based on topology and its consistency with the longitudinal and lateral shower profiles. The HF PMT hits are characterized by a large apparent energy deposit in long (short) fibers and very little or no energy in short (long) fibers within a single HF tower. Therefore, a simple energy ratio

$$R = \frac{E_\mathrm{L} - E_\mathrm{S}}{E_\mathrm{L} + E_\mathrm{S}}, \tag{1}$$

where E_L and E_S are the energies of the long and short fiber RecHits, respectively, in a given HF tower, can be readily used to identify the HF PMT hits. The HF PMT hits in the long fibers will have $R \approx 1$ and those in the short fibers will have $R \approx -1$. Since the short fibers start at a depth of approximately 22 cm from the front face of the detector and based on the longitudinal shower profile, it is very unlikely that a particle deposits all of its energy only in the short fibers. Therefore, the R ratio is expected to have a strong discriminating power to identify the HF PMT hits in the short fibers. On the other hand, since the long fibers extend through the entire depth of the HF starting from the front face, it is possible that some particle, in particular photons and electrons, deposit all or most of their energy in the long fibers. Therefore, a filtering algorithm based only on the R ratio will also reject some real energy. One such algorith will be discussed in more detail in Section 4. In order to make the filtering algorithms safer for the real energy, one possibility is to take into account the lateral shower profile and incorporate some type of isolation variable into the filtering algorithm. One such algorithm will be discussed in more detail in Section 5.

3 Data and Monte Carlo Samples

Several different data samples were used to study and optimize the performance of the HF PMT hit cleaning algorithms. For collison events at $\sqrt{s} = 0.9$ and 2.36 TeV collected during the LHC startup period at the end of 2009 the following skimmed dataset was used:

- /MinimumBias/BeamCommissioning09-BSCNOBEAMHALO-Feb9Skim_v1/RAW-RECO

In order to select the collision events from the above dataset the following event selection was applied:

1. BPTX technical trigger (TT) bit 0 fired — indicates timing consistent with two proton bunches crossing in the center of CMS
2. At least one of the BSC MinBias TT bits (40 OR 41) fired— this requirement was already applied in the above dataset in order to skim events from the MinimumBias primary dataset

3. None of the BSC beam halo TT bits (36 OR 37 OR 38 OR 39) fired — this requirement was already applied in the above dataset in order to skim events from the MinimumBias primary dataset

4. At least one good primary vertex present in the event

5. Removal of the so-called scraping events

6. HLT_PhysicsDeclared bit set — indicates that both LHC beams were stable and all CMS sub-detectors were operating without problems

7. Only events from the *good runs* were considered

This event selection is identical to the one used in the calorimeter $\not\!\!E_{\mathrm{T}}$ commissioning note [1] where a more detailed description can be found. The majority of the pp collisions contained in the above dataset are collisions at $\sqrt{s} = 0.9$ TeV. The total of 180649 events at $\sqrt{s} = 0.9$ TeV and 10339 events at $\sqrt{s} = 2.36$ TeV pass the above selection.

In addition to the above collision dataset, the following Monte Carlo simulation datasets were used:

- /MinBias/Summer09-V16D_900GeV-v1/GEN-SIM-RECO
  - For comparison with the collison data, the event selection requirements 2, 4, and 5 were applied to events contained in this dataset
  - 651308 events passed the selection
- /MinBias/Summer09-V16E_2360GeV-v1/GEN-SIM-RECO
  - For comparison with the collison data, the event selection requirements 2, 4, and 5 were applied to events contained in this dataset
  - 559464 events passed the selection
- /MinBias/Summer09-MC_31X_V3_7TeV-v1/GEN-SIM-RECO
  - The event selection requirements 4 and 5 were applied to events contained in this dataset (BSC triggers were not properly simulated in this dataset)
  - 581929 events passed the selection
- /QCD_Pt80/Summer09-MC_31X_V3_7TeV-v1/GEN-SIM-RECO
  - No event selection was applied to events contained in this dataset
  - Total of 1M events analyzed
- /QCDFlat_Pt15to3000/Summer09-MC_31X_V9_7TeV-v1/GEN-SIM-RECO
  - No event selection was applied to events contained in this dataset
  - Total of 908872 events analyzed
- /SinglePhotonsInHF_Flat_E10to500GeV/ferencek-SinglePhotonsInHF_Flat_E10to500GeV -789e70cb4057095ca9760d72f15eb0de/USER
  - This is a privately produced sample of back-to-back photon pairs flat in ϕ, energy (from 10 to 500 GeV) and η fired only into the HF acceptance. The dataset is published in `cms_dbs_caf_analysis_01` local DBS instance
  - Total of 1M events produced and analyzed

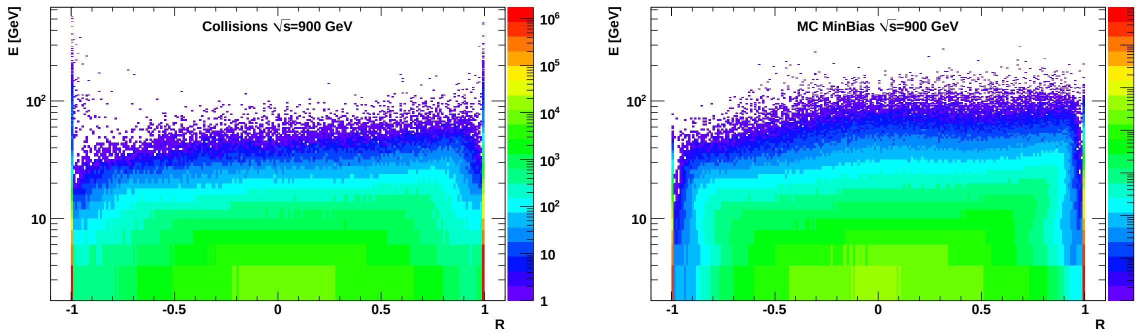

Figure 1: HF RecHit energy vs. R ratio for long and short fiber RecHits in 900 GeV collision data (left) and 900 GeV MinBias Monte Carlo simulation (right). Entries with energy around or above 100 GeV and $R \approx 1$ in the left plot are the long fiber PMT hits and those with $R \approx -1$ are the short fiber PMT hits.

4 PET Cleaning Algorithm

As already discussed in Section 2, a simple energy ratio defined in Eq. 1 can be readily used to identify the HF PMT hits. Figure 1 illustrates the fact that the HF PMT hits in the long fibers have $R \approx 1$ and those in the short fibers have $R \approx -1$. From these plots it can also be noticed that most of the HF PMT hits have energies around 100 GeV.

From the plots in Figure 1 it is clear that a simple cut on the R ratio is not sufficient to identify the HF PMT hits and not reject any real energy. Therefore, an additional energy cut has to be applied, in addition to a cut on the R ratio. Figures 2 and 3 show the energy spectra of the long and short fiber RecHits with $R > 0.98$ and $R - 0.98$, respectively, in different $i\eta$ rings in 900 GeV collision data and 900 GeV MinBias Monte Carlo simulation. These plots indicate that the RecHit energy spectrum changes from one $i\eta$ ring to another. Therefore, an optimal energy cut should be $i\eta$-dependent. The cut is put at the highest energy for which data and MC still have equal rates and beyond which the HF PMT hits start to dominate. In this way a compromise is achieved between identifying as many real HF PMT hits as possible and having a minimum impact on the real energy.

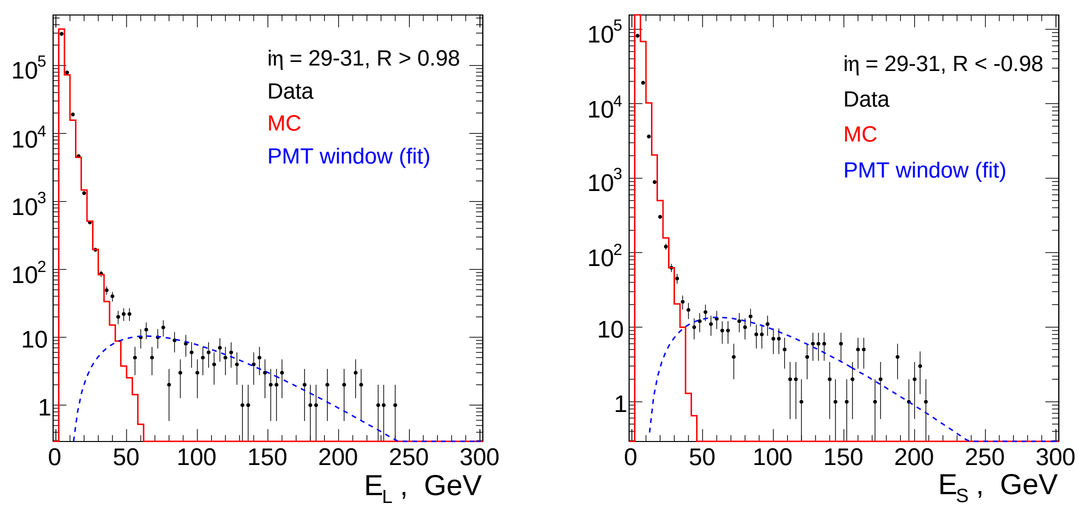

Figure 2: Energy spectra for the long (left) and short (right) fiber RecHits with $R > 0.98$ and $R - 0.98$, respectively, in $i\eta$ rings 29–31 in 900 GeV collision data and 900 GeV MinBias Monte Carlo simulation. The Monte Carlo energy spectrum is nomalized to the real data in the 20–40 GeV range. The same procedure was also followed for the short fiber RecHits. The blue dashed line represents the PMT hit energy spectrum obtained after subtracting the Monte Carlo distribution from the real data distribution.

However, the energy thresholds determined in this way are not used directly but a second order polynomial in $i\eta$ is fitted to these values and is used to parameterize the energy threshold as a function of $i\eta$, separately for long and short fiber RecHits. Both polynomial energy threshold (PET) parameterizations are shown in Figure 4.

Finally, the PET cleaning algorithm is defined as follows:

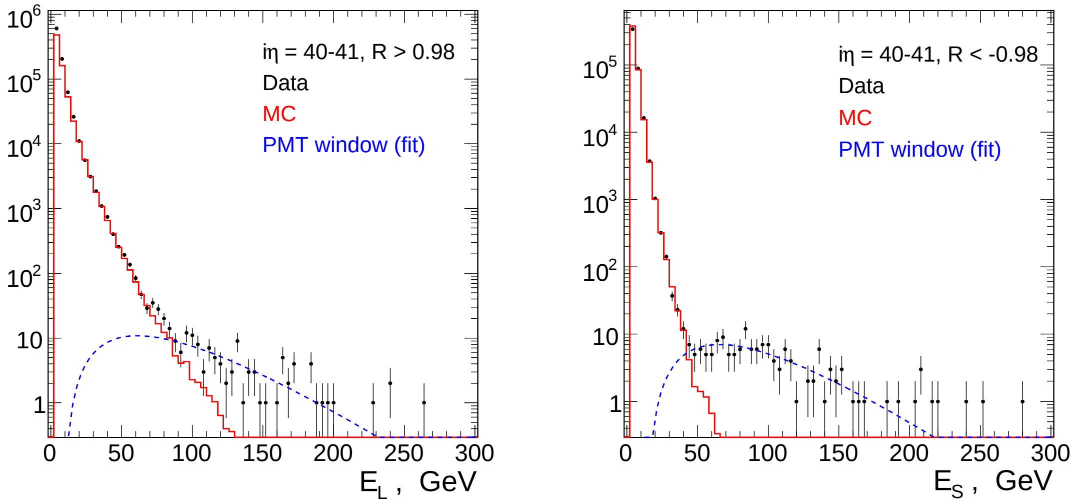

Figure 3: Energy spectra for the long (left) and short (right) fiber RecHits with $R > 0.98$ and $R - 0.98$, respectively, in $i\eta$ rings 40–41 in 900 GeV collision data and 900 GeV MinBias Monte Carlo simulation. The Monte Carlo energy spectrum is nomalized to the real data in the 20–40 GeV range. The same procedure was also followed for the short fiber RecHits. The blue dashed line represents the PMT hit energy spectrum obtained after subtracting the Monte Carlo distribution from the real data distribution.

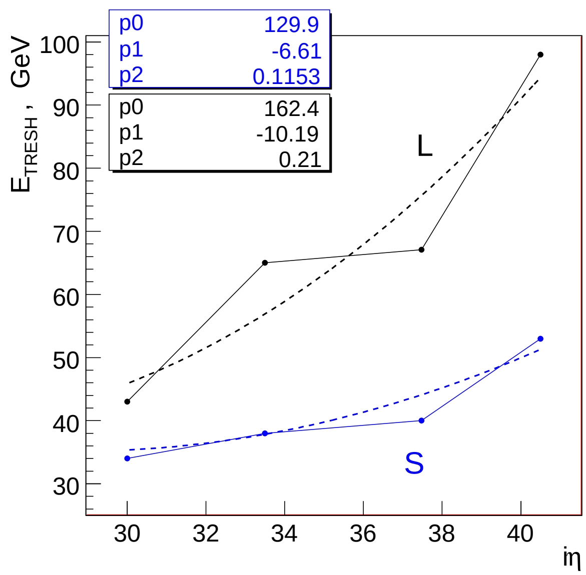

Figure 4: Polynomial energy threshold (PET) parameterization for the long and short fiber RecHits.

- Long fiber RecHits are flagged if: $R > 0.98$ AND $E_{\mathrm{L}} > E_{\mathrm{PET,L}}(i\eta) = 162.4 - 10.19|i\eta| + 0.21(i\eta)^2$
- Short fiber RecHits are flagged if: $R < -0.98$ AND $E_{\mathrm{S}} > E_{\mathrm{PET,S}}(i\eta) = 129.9 - 6.61|i\eta| + 0.1153(i\eta)^2$

Performance of the PET cleaning algorithm is presented in Section 6.

5 S9/S1 Cleaning Algorithm

In order to make the PET cleaning algorithm described in Section 4 safer for the real energy deposits in the long fibers, an isolation variable was introduced that takes into account the lateral shower profile and ensures that the real energy deposits, that are typically non-isolated, are not flagged as potential HF PMT hits. An isolation variable that was introduced for this purpose is S9/S1 and for the long fiber RecHits it is defined as

$$\left(\frac{\mathrm{S9}}{\mathrm{S1}}\right)_{\mathrm{L}} = \frac{E_{\mathrm{S}} + \sum_{i=1}^{4} E_{\mathrm{L},i} + \sum_{i=1}^{4} E_{\mathrm{S},i}}{E_{\mathrm{L}}}, \qquad (2)$$

where E_{L} and E_{S} are the energies of the long and short fiber RecHits, respectively, and the two sums in the above expression go over the four neighboring towers that share an edge with the towers for which the S9/S1 variable is being calculated as illustrated in Figure 5. Similar variable was first introdced by the PFlow group [2].

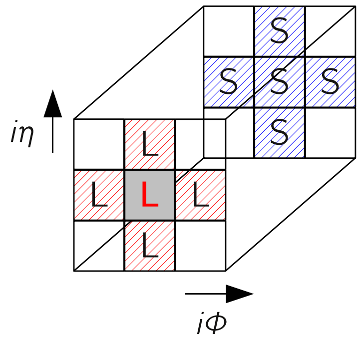

Figure 5: A schematic drawing of the grid of nine HF towers highlighting those used to calculate the S9/S1 isolation variable.

The S9/S1 cleaning algorithm is an extension of the PET algorithm. It is only applied to the long fiber RecHits and it uses the same energy threshold parameterization as the PET algorithm. However, instead of using the R ratio to identify the HF PMT hits it uses the S9/S1 isolation variable. Hence, only those RecHits that are above the $E_{\mathrm{PET,L}}(i\eta)$ energy threshold and have the S9/S1 isolation below a certain value are flagged as potential PMT hits. An exception is the $i\eta$ ring 39 which is located behind HE and receives very little energy. Therefore, it is safe to use the PET algorithm for both long and short fiber RecHits in $i\eta$ ring 39. Since the lateral shower size is expected to roughly scale with $\ln E$, the S9/S1 isolation is allowed to scale linearly with $\ln E$. A 2D distribution of the S9/S1 isolation vs. energy for long fiber RecHits is shown in Figure 6. A narrow strip of isolated hits in the lower right corner present in collision data but not in the Monte Carlo simulation are the HF PMT hits.

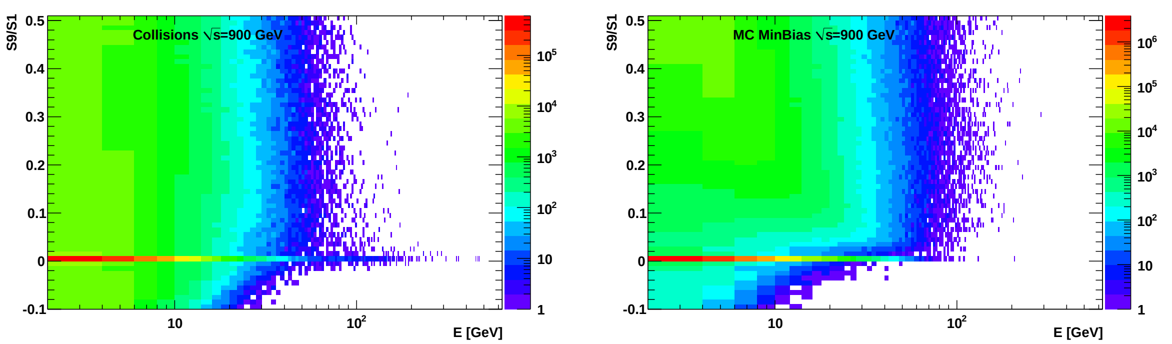

Figure 6: S9/S1 vs. energy for long fiber RecHits in 900 GeV collision data (left) and 900 GeV MinBias Monte Carlo simulation (right). A narrow strip of isolated hits in the lower right corner present in collision data but not in the Monte Carlo simulation are the HF PMT hits.

The S9/S1 cut line is defined for each $i\eta$ ring separately has the following dependence

$$\mathrm{S9/S1}(E) = k \ln E + l, \tag{3}$$

where k is the slope and l is the y-intercept, and is required to cross the x-axis at the energy value defined by the $E_{\mathrm{PET,L}}(i\eta)$ energy threshold, i.e.,

$$l = -k \ln(E_{\mathrm{PET,L}}(i\eta)). \tag{4}$$

All that is left to be optimized is the slope k. In order to make the S9/S1 cleaning algorithm as safe as possible for the real energy, the slope k was optimized using the single photons sample described in Section 3 in such a way that 0.1% of the RecHits above the $E_{\mathrm{PET,L}}(i\eta)$ energy threshold are flagged as potential PMT hits. An example of such optimization for $i\eta$ ring 35 along with the optimized slopes for all $i\eta$ rings are shown in Figure 7. The optimized slopes increase as a function of $i\eta$ which is expected since the physical size of the HF towers descreases as $i\eta$ increases making photons less isolated in higher $i\eta$ rings. The last three $i\eta$ rings, however, do not follow the same trend as $i\eta$ rings 30 through 38. This is caused by the change in the ϕ-segmentation of the last two $i\eta$ rings making towers in $i\eta$ ring 39 appear less isolated due to the bigger neighboring towers in $i\eta$ ring 40. Towers in $i\eta$

ring 41, on the other hand, appear more isolated simply because they are located on the edge of HF and have only three neighboring towers. As shown in the right plot in Figure 7, the optimized slopes in $i\eta$ rings 30 through 38 are parameterized using a second order polynomial while those in the last three $i\eta$ rings are used directly as they come from the optimization procedure. The final values of the optimized slopes k for $i\eta$ rings 30 though 41 are the following:

- 0.0164905, 0.0238698, 0.0321383, 0.041296, 0.0513428, 0.0622789, 0.0741041, 0.0868186, 0.100422, 0.135313, 0.136289, 0.0589927.

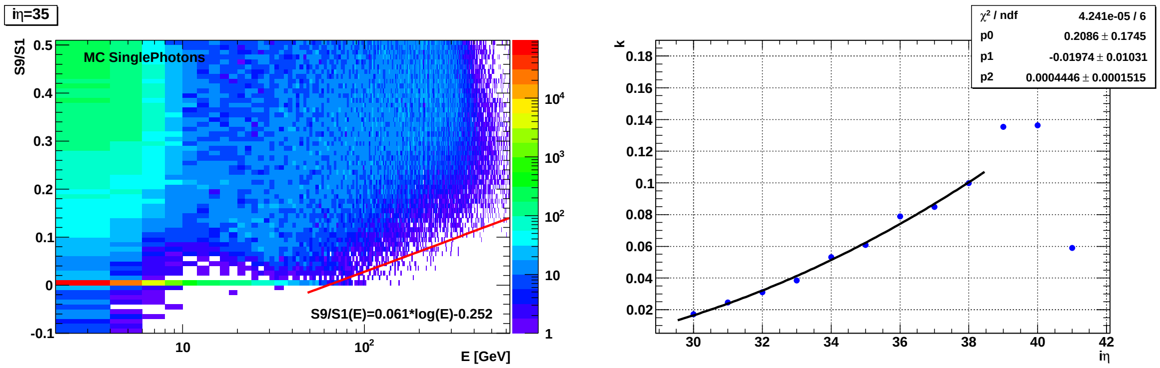

Figure 7: Optimized S9/S1 cut line for $i\eta$ ring 35 in single photons sample (left) and optimized slopes for all $i\eta$ rings (right).

Performance of the S9/S1 cleaning algorithm is presented in Section 6.

6 Performance of PET and S9/S1 Algorithm

In this section we present the performance of the PET and S9/S1 algorithm in 900 GeV and 2.36 TeV collision data. In addition effects of the noise cleaning algorithms on the simulated data are presented.

6.1 Performance on Short Fiber RecHits

For the short fiber RecHits only the PET algorithm is applied. Figures 8 and 9 show the energy and E_{T} spectra of the short fiber RecHits, all and those flagged by the PEt algorithm, in 900 GeV and 2.36 TeV collision data, respectively. It can be noticed that the PET algorithm efficiently flags almost all high energy hits in both 900 GeV and 2.36 TeV collision data.

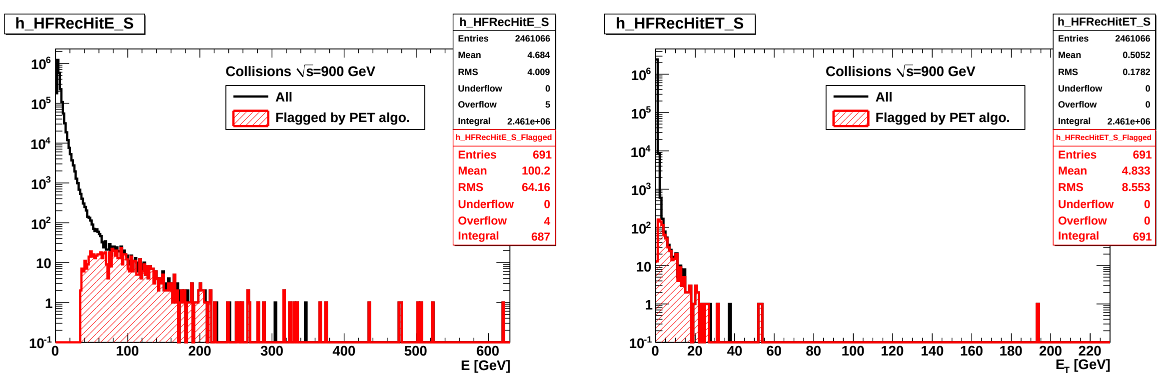

Figure 8: Energy and E_{T} spectra of the short fiber RecHits, all and those flagged by the PET algorithm, in 900 GeV collision data.

Figure 10 shows the performance of the PET algorithm on different Monte Carlo samples. The goal here is that the noise cleaning algorithm flags as little real energy as possible. This is determined by looking at the fraction of RecHits with $E_{\mathrm{T}} > 5$ GeV that are flagged in Monte Carlo samples. The final results are summarized in Table 1 and show that the PET cleaning algorithm is safe for real energy deposits in short fibers.

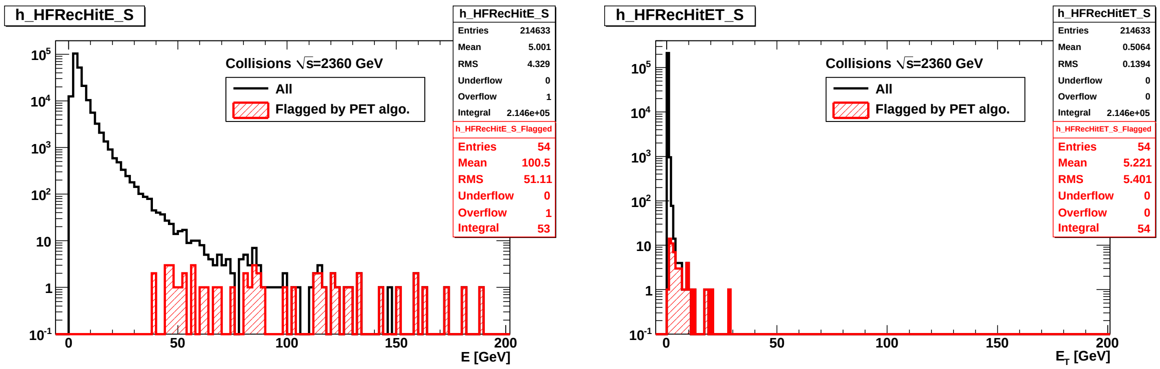

Figure 9: Energy and E_{T} spectra of the short fiber RecHits, all and those flagged by the PET algorithm, in 2.36 TeV collision data.

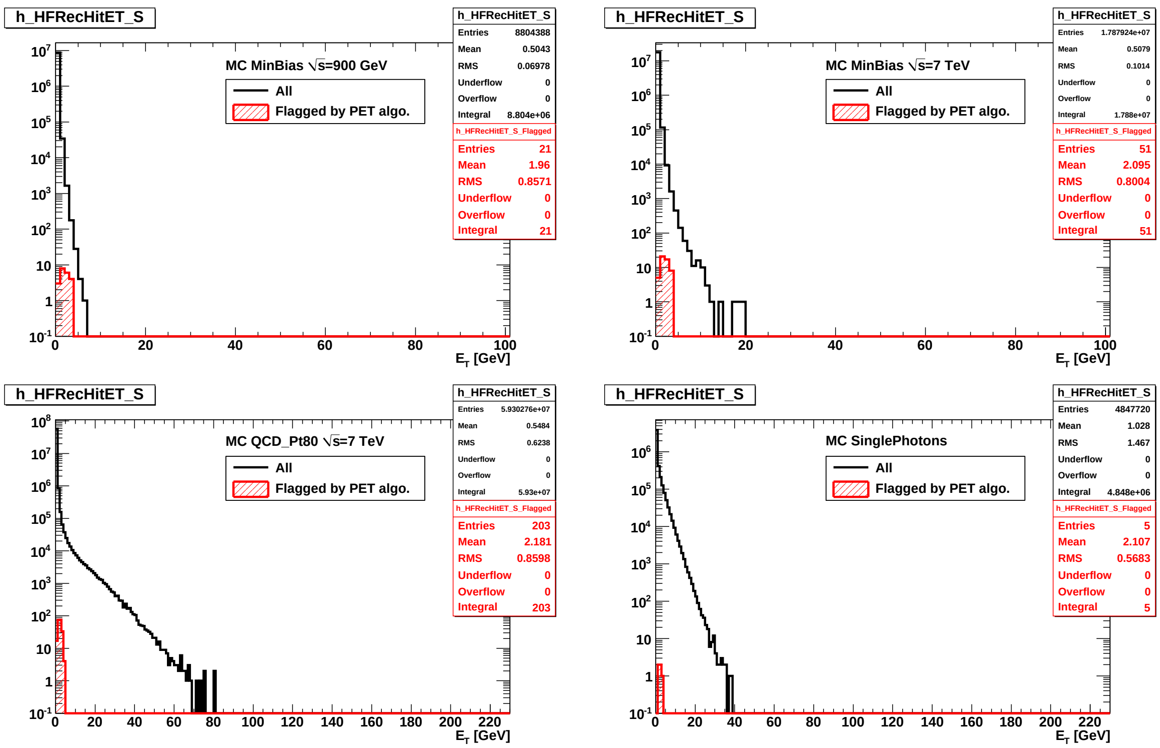

Figure 10: E_{T} spectra of the short fiber RecHits, all and those flagged by the PET algorithm, in different Monte Carlo samples.

Table 1: Fraction of short fiber RecHits with $E_{\mathrm{T}} > 5$ GeV flagged by the PET algorithm in different Monte Carlo samples.

MC sample	Fraction of flagged RecHits with $E_{\mathrm{T}} > 5$ GeV
900 GeV MinBias	$0/5$
7 TeV MinBias	$0/275$
7 TeV QCD_Pt80	$0/128578$
Single Photons	$0/147187$

6.2 Performance on Long Fiber RecHits

For the long fibers RecHits either PET or S9/S1 algorithm can applied to clean the HF PMT hits. Figures 11 and 12 show the energy and E_{T} spectra of the long fiber RecHits, all and those flagged by the PET and S9/S1 algorithm, in 900 GeV and 2.36 TeV collision data, respectively. It can be noticed that both algorithms efficiently flag high energy hits in both 900 GeV and 2.36 TeV collision data. Overall, the PET cleaning algorithm is slighlty more efficient at identifying the PMT hits. Nevertheless, for the highest E_{T} hits both algorithms show almost identical peformance.

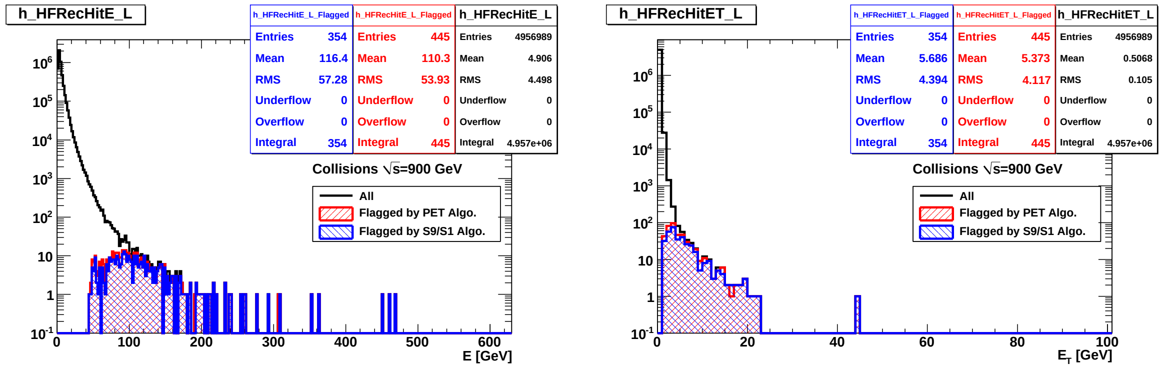

Figure 11: Energy and E_{T} spectra of the long fiber RecHits, all and those flagged by the PET and S9/S1 algorithm, in 900 GeV collision data.

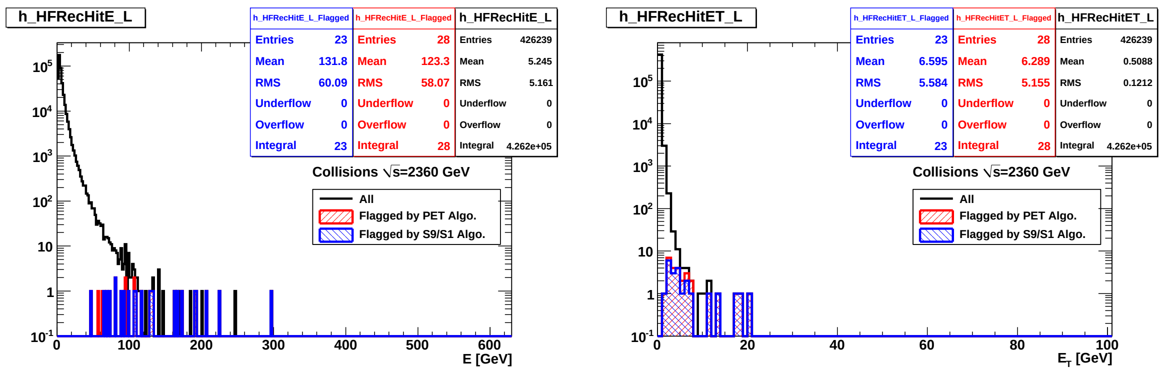

Figure 12: Energy and E_{T} spectra of the long fiber RecHits, all and those flagged by the PET and S9/S1 algorithm, in 2.36 TeV collision data.

Figure 13 shows the performance of the PET and S9/S1 algorithm on different Monte Carlo samples. The goal here as before is that the noise cleaning algorithms flag as little real energy as possible. This is determined by looking at the fraction of RecHits with $E_{\mathrm{T}} > 5$ GeV that are flagged in Monte Carlo samples. The final results are summarized in Table 2 and show that the S9/S1 cleaning algorithm is safer for real energy deposits in long fibers than the PET algorithm.

Table 2: Fraction of long fiber RecHits with $E_{\mathrm{T}} > 5$ GeV flagged by the PET and S9/S1 algorithm in different Monte Carlo samples.

	Fraction of RecHits with $E_{\mathrm{T}} > 5$ GeV flagged by	
MC sample	PET algorithm	S9/S1 algorithm
900 GeV MinBias	$2/46 \approx 4.3\%$	$0/46$
7 TeV MinBias	$30/1346 \approx 2.2\%$	$3/1346 \approx 0.22\%$
7 TeV QCD_Pt80	$704/233138 \approx 0.30\%$	$32/233138 \approx 0.014\%$
Single Photons	$4454/829530 \approx 0.54\%$	$991/829530 \approx 0.12\%$

Based on the performance of the PET and S9/S1 algorithm on the long and short fiber RecHits it was decided to use the S9/S1+PET combination as the default HF cleaning algorithm; the S9/S1 algorithm for the long fiber RecHits and the PET algorithm for the short fiber RecHits.

6.3 Impact on Higher Level Objects

Anomalous signals in HF can have a detrimental effect on higher level physics object such as jets and missing transverse energy ($\not{E}_{\mathrm{T}}$). The $\not{E}_{\mathrm{T}}$ is in particular affected by the HF PMT hits. This is understandable since the PMT hits create an apparent energy imbalance for events in which they appear. The energy spectrum of the HF PMT hits is peaked at around 100 GeV and considering the location of the HF towers at high pseudorapidities, most of the HF PMT hits do not create a very large $\not{E}_{\mathrm{T}}$. Nevertheless, in $\sqrt{s} = 900$ GeV and 2.360 TeV collision data they were one of the major sources of the $\not{E}_{\mathrm{T}}$ tails.

The main goal of the noise cleaning algorithm is to clean the higher level object from any anomalous signals and

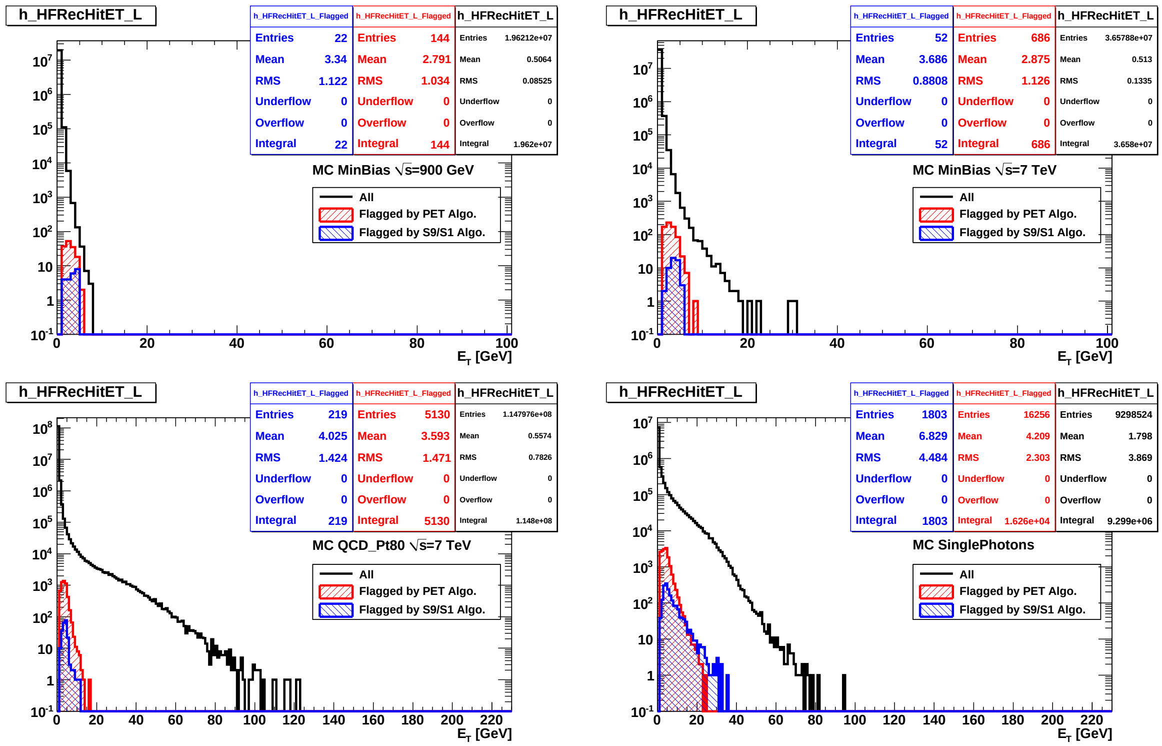

Figure 13: E_{T} spectra of the long fiber RecHits, all and those flagged by the PET and S9/S1 algorithm, in different Monte Carlo samples.

at the same time have a negligible effect on the real energy. Figure 14 shows the $\not\!\!E_{\mathrm{T}}$ formed only from the energy deposits in HF before and after applying the PET and S9/S1+PET cleaning algorithms. As can be seen, both cleaning algorithms almost completely clean up the $\not\!\!E_{\mathrm{T}}$ tails. However, some entries remain in the $\not\!\!E_{\mathrm{T}}$ tails even after applying the noise cleanup. These entires are due to double hits where both long and short fibers in the same HF tower have appreciable energies but are otherwise isolated. Because of that, such hits are not identified by either PET or S9/S1 algorithm.

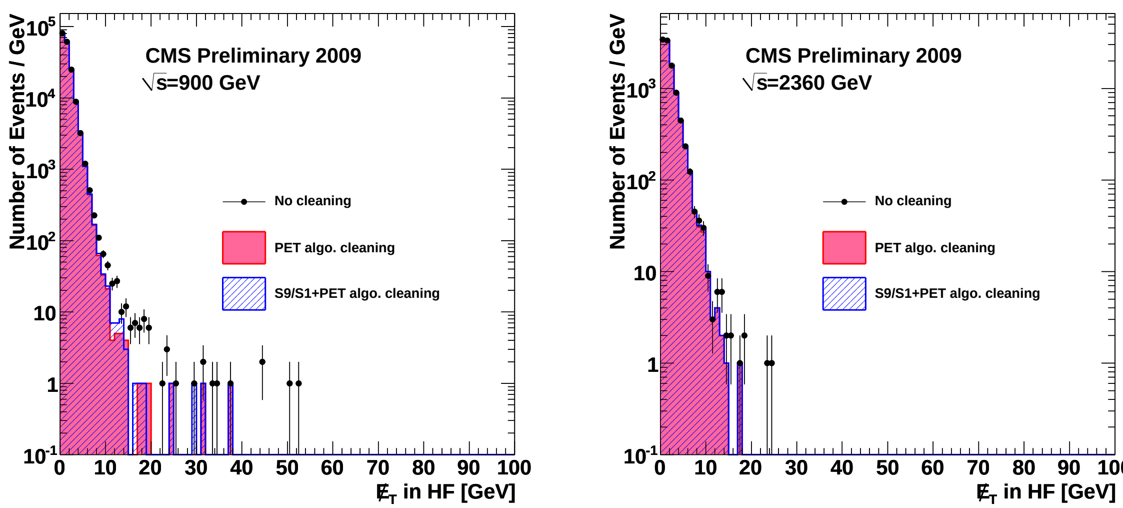

Figure 14: $\not\!\!E_{\mathrm{T}}$ in HF before and after applying the PET and S9/S1+PET cleaning algorithm.

Another important aspect of the noise cleanup is that it improves the agreement between real data and Monte Carlo simulation. Figure 15 shows the $\not\!\!E_{\mathrm{T}}$ in HF distribution in $\sqrt{s} = 900$ GeV and 2.360 TeV collision data compared with Monte Carlo simulation.

In order to study the impact of the two cleaning algorithms on the higher level objects and demonstrate that they

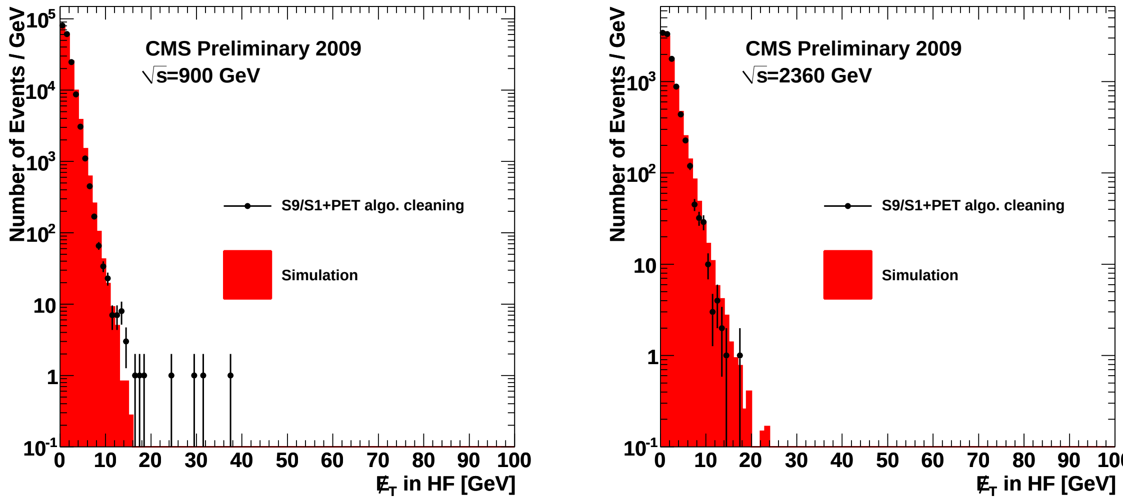

Figure 15: $\not{E}_T$ in HF after applying the S9/S1+PET cleaning algorithm in $\sqrt{s} = 900$ GeV (left) and 2.360 TeV (right) collision data compared with Monte Carlo simulation.

are safe for real energy, a reflagger tool described in Section 8 was used on QCD_Pt80 Monte Carlo sample listed in Section 3. Two new HF RecHit collections were created using the reflagger tool from the original HF RecHit collection, one with PET and the other with S9/S1+PET algorithm applied. From these new HF RecHit collections and other intact ECAL and HCAL RecHit collections tow sets of jets and $\not{E}_T$ were re-reconstructed, one for each cleaning algorithm. Figure 16 shows the final impact of the two cleaning algorithms on some of the jet distributions for jets in HF. The impact on the noise cleaning algorithm on the $\not{E}_T$ reconstruction is shown in Figure 17. These figures demonstrate that both noise cleaning algorithms have a negligible effect on the jet and $\not{E}_T$-related quantities with S9/S1+PET combination being slightly safer.

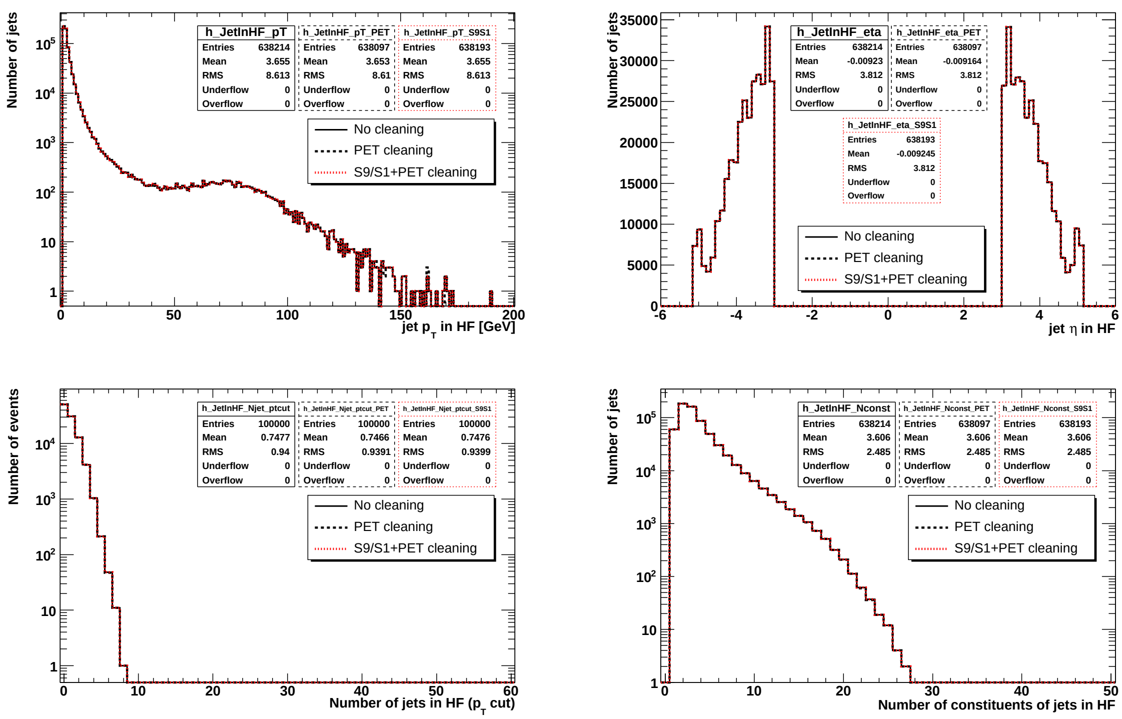

Figure 16: Impact of the noise cleaning algorithms on p_T, η, jet multiplicity for jets with $p_T > 5$ GeV, and number of constituents distributions for jets in HF in QCD_Pt80 Monte Carlo sample.

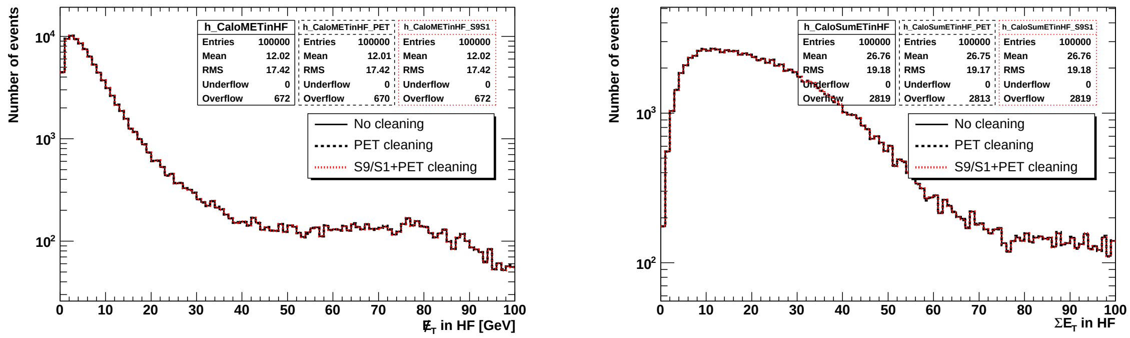

Figure 17: Impact of the noise cleaning algorithms on $\not\!\!E_{\mathrm{T}}$ and $\sum E_{\mathrm{T}}$ in HF in QCD_Pt80 Monte Carlo sample.

7 Rate of HF PMT Hits at $\sqrt{s} = 7$ TeV

One of the characteristics of the HF PMT hits noticed in 900 GeV and 2.36 TeV collision data is that their rate per event roughly scales linearly with the amount of energy deposited in HF, excluding the energy of the PMT hits. This is shown in Figure 18 for combined 900 GeV and 2.36 TeV collision data. Based on this linear dependence and the mean $\sum E$ in HF coming from Monte Carlo simulation of Minimum Bias events at 7 TeV, one can roughly predict the rate of PMT hits in Minimum Bias events at 7 TeV. Based on the predicted mean $\sum E$ in HF for Minimum Bias events at 7 TeV shown in Figure 19, one would expect the average rate of HF PMT hits of 12.2×10^{-3} hits/event in the Minimum Bias data at 7 TeV.

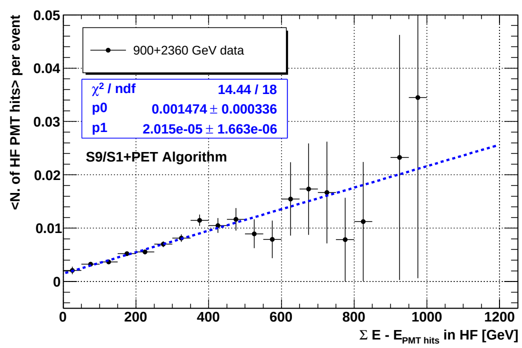

Figure 18: Average number of the HF PMT hits as a function of $\sum E$ in HF excluding the energy of the PMT hits in the combined 900 GeV and 2.36 TeV collision data.

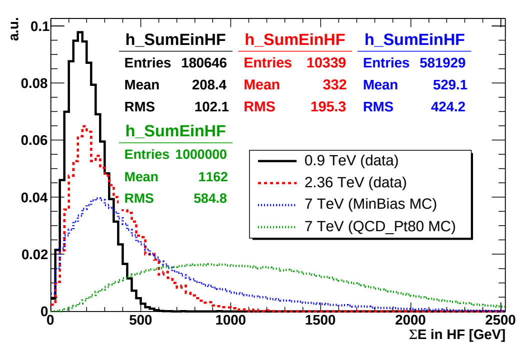

Figure 19: Distribution of $\sum E$ in HF in 900 GeV and 2.36 TeV collision data and MinBias and QCD_Pt80 Monte Carlo simulated events at 7 TeV.

The actual observed rate of the HF PMT hits in the first 7 TeV collision data coming from run 132440 was $(13.2 \pm 0.3) \times 10^{-3}$ hits/event which is relatively close to the predicted value. The rate of PMT hits per event as a function of $\sum E$ in HF, excluding the energy of the PMT hits, for run 132440 is shown in Figure 20. This linear dependence is consistent with the one observed in the combined 900 GeV and 2.36 TeV collision data.

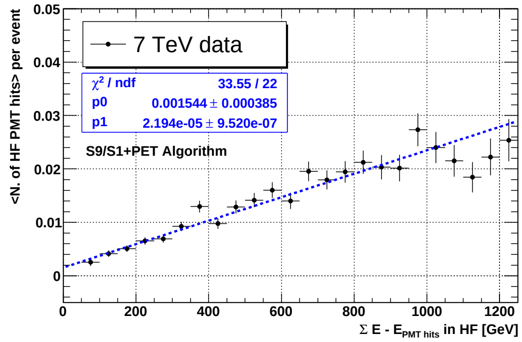

Figure 20: Average number of the HF PMT hits as a function of $\sum E$ in HF excluding the energy of the PMT hits in the first 7 TeV collision data (run 132440).

8 CMSSW Tools for Noise Cleaning

The HF noise flags described above are set in the rechits as part of the default reconstruction beginning with CMSSW release 3_7_0 (VERIFY ONCE OFFICIAL!). These revised flags can also be also be applied when re-reconstructing data within a CMSSW_3_5 release by first checking out the following package versions:

- cvs co -r V00-00-07 RecoLocalCalo/HcalRecProducers
- cvs co -r V00-00-03 RecoLocalCalo/HcalRecAlgos

and then running the standard reconstruction process.

Instead of performing a full re-reconstruction of the data to make use of the information in these flags, one may instead make use of a “reflagger” tool, which creates new collections of rechits with a revised set of flags. The reflagger can produce rechits with flags set according to the HF noise algorithm described in previous sections, or it can set rechit flags based on users’ private algorithms. This allows for the possibility of analysis-specific optimization of noise algorithms for the rechit collections.

We present here instructions on how to use the reflagger on existing reconstructed data to produce a new set of rechits and CaloTowers with new flag information. There are three steps involved in this process: setting up the reflagger, altering the severity level computer to make use of the new flags, and creating a configuration file to produce the new rechit and calotower collections.

8.1 Setting Up the Reflagger

Check out and compile the reflagger via:

- cd <CMSSW_RELEASE_VERSION>/src
- cvs co -r V00-00-07 RecoLocalCalo/HcalRecProducers
- cvs co -r V00-00-03 RecoLocalCalo/HcalRecAlgos
- cp RecoLocalCalo/HcalRecAlgos/test/hcalrechitreflagger_cfi.py RecoLocalCalo/HcalRecAlgos/python/
- scram b -j4

The reflagging code is contained within RecoLocalCalo/HcalRecAlgos/test/HcalRecHitReflagger.cc. This code loops over the initial rechit collection, calculates a new flag based on the algorithms defined in the code, and produces a new rechit collection with the results of the new flag stored in flag bit ’UserDefinedBit0’. (A full list of rechit flags is provided in https://twiki.cern.ch/twiki/bin/view/CMS/HcalRecHitFlagAssignments.) The configuration file RecoLocalCalo/HcalRecAlgos/test/hcalrechitreflagger_cfi.py provides a number of options for how this new flag bit is set:

- hfAlgo3test: this flags rechits according to the combination PET/S9S1 algorithm described above.

- hfAlgo2test: this flags rechits using only the PET algorithm.
- hfBitAlwaysOn: this sets the UserDefinedBit0 bit to 'on' for all rechits.
- hfBitAlwaysOff: this sets the UserDefinedBit0 bit to 'off' for all rechits.

In general, one should enable only one of these options, unless one understands how the options work in combination with each other. By default, the hfAlgo3test option is the only option enabled.

8.2 Altering the Severity Level Computer

Once the reflagger has been set up to the desired specifications, the severity level computer must be altered to make use of the new UserDefinedBit0 flag. In the current example, the UserDefinedBit0 is used to override the functionality of the existing HF noise flag (labelled "HFLongShort"). Thus, the severity level must be modified in two ways: UserDefinedBit0 must be appended to the computer, with a level high enough so that rechits flagged with this bit are excluded from CaloTower creation, and HFLongShort must be removed from the computer, so that the results of this flag are ignored by the CaloTower creator.

When creating new individual flags, it is generally enough to simply add those flags to the severity level computer, with a severity level high enough to provide the desired functionality within the CaloTower creator. (Rechits with severity level greater of at least 10 are excluded from CaloTowers, while those with severity level between 1-9 are included in CaloTowers, but marked as "problematic"). Examples of code to both add and remove rechit flags from the severity level computer are provided in Appendix A.

8.3 Producing New Rechits and CaloTowers

After the rechit reflagger and severity level computer have been altered to one's preferred specifications, all that remains is to create a configuration file that makes use of these tools to create new rechit and CaloTower collections. A sample configuration file is shown in Appendix B. This configuration file uses the new rechits and CaloTowers to create new jet and MET collections.

Variables of special interest within this cfg file are:

- process.hcalRecAlgos – This is the severity level computer. It is modified by calls to the methods "RemoveFlag" and "AddFlag" described in Appendix A. For this reason, the file "RemoveAddSevLevel.py" shown in Appendix A must be included in the same directory as this cfg file.
- process.hfrecoReflagged – This is the name of the collection of reflagged rechits.
- process.towerMaker.hfInput and process.towerMakerWithHO.hfInput – These variables store the names of the HF rechit collections used as inputs to the CaloTower creator. Both must be set to "hfrecoReflagged" in order to use the reflagged rechits.
- process.rereco_step – This defines the reconstruction (jets, MET, etc.) to be performed using the reflagged CaloTowers.

9 Conclusions

10 Acknowledgments

We thank the technical and administrative staff at CERN and the other CMS Institutions.

References

[1] K. Hatakeyama et al., "Commissioning of Uncorrected Calorimeter Missing Transverse Energy in Zero Bias and Minimum Bias Events at $\sqrt{s} = 900$ GeV and $\sqrt{s} = 2360$ GeV," *CMS AN* **2009/029** (2009).

[2] P. Janot, "Early Data Cleaning in PF Reconstruction," *http://indico.cern.ch/getFile.py/access?contribId=5&resId=0&materialId=slides&confId=76306*.

A Removing and Adding Flags from Severity Level Computer

This code may also be downloaded directly from the twiki at https://twiki.cern.ch/twiki/pub/CMS/HcalRecHitReflagger/RemoveAddSevLevel.py.txt.

```
import FWCore.ParameterSet.Config as cms

def RemoveFlag(SLComp,flag="HFLongShort"):
    ’’’ Removes the specified flag from the Severity Level Computer,
    and returns the revised Computer.’’’

    REMOVE=-1 # Track which Severity Level has been modified

    # Loop over all levels
    for i in range(len(SLComp.SeverityLevels)):
        Flags=SLComp.SeverityLevels[i].RecHitFlags.value()
        if flag not in Flags:  # Flag not present for this level
            continue
        #Remove flag
        Flags.remove(flag)
        ChanStat=SLComp.SeverityLevels[i].ChannelStatus.value()
        # Check to see if Severity Level
        #no longer contains any useful information
        if len(Flags)==0 and ChanStat==[’’]:
            REMOVE=i
        else:
            # Set revised list of flags for this severity level
            SLComp.SeverityLevels[i].RecHitFlags=Flags
        break

    # Removing flag results in empty severity level; remove it
    if (REMOVE>-1):
        SLComp.SeverityLevels.remove(SLComp.SeverityLevels[REMOVE])

    return SLComp

####################################################################

def AddFlag(SLComp,flag="UserDefinedBit0",severity=10):
    ’’’ Adds specified flag to severity level computer using
        specified severity level.
        If flag already exists at another severit level,
        it is removed from that level.
        ’’’

    AddedSeverity=False
    REMOVE=-1
    #Loop over severity Levels
    for i in range(len(SLComp.SeverityLevels)):
        Level=SLComp.SeverityLevels[i].Level.value()
        Flags=SLComp.SeverityLevels[i].RecHitFlags.value()
        if Level==severity:  # Found the specified level
            if (Flags==[’’]):
                Flags=[flag] # Create new vector for this flag
            else:
                Flags.append(flag) # append flag to existing vector
            # Set new RecHitFlags vector
            SLComp.SeverityLevels[i].RecHitFlags=Flags
```

```
                AddedSeverity=True
        else:  # Found some other level; be sure to remove flag from it
            if flag not in Flags:
                continue
            else:
                Flags.remove(flag)
                # Removing flag leaves nothing else:
                # need to remove this level completely
                if len(Flags)==0 and ChanStat==['']:
                    REMOVE=i
                else:
                    SLComp.SeverityLevels[i].RecHitFlags=Flags

    # Remove any newly-empty levels
    if (REMOVE>-1):
        SLComp.SeverityLevels.remove(SLComp.SeverityLevels[REMOVE])

    # No existing severity level for specified severity was found;
    # add a new one
    if (AddedSeverity==False):
        SLComp.SeverityLevels.append(cms.PSet(Level=cms.int32(severity),
                                              RecHitFlags=cms.vstring(flag),
                                              ChannelStatus=cms.vstring("")))
    return SLComp


def PrintLevels(SLComp):
    print ''Severity Level Computer Levels and'',
    print '' associated flags/Channel Status values:''
    for i in SLComp.SeverityLevels:
        print ''\t Level = %i''%i.Level.value()
        print ''\t\t RecHit Flags = %s''%i.RecHitFlags.value()
        print ''\t\t Channel Status = %s''%i.ChannelStatus.value()
        print
    return

```

B Sample .cfg File for Generated Reflagged RecHits and CaloTowers

This example cfg file will produce a new set of rechits and CaloTowers using the default S9S1/PET algorithm described within this note to reflag rechits. This example may also be found at the twiki page https://twiki.cern.ch/twiki/pub/CMS/HcalRecHitReflagger/SampleCfgFile.py.txt.

```

import FWCore.ParameterSet.Config as cms

process = cms.Process(’USER’)

# import of standard configurations
process.load(’Configuration/StandardSequences/Services_cff’)
process.load(’FWCore/MessageService/MessageLogger_cfi’)
process.load(’Configuration/StandardSequences/GeometryExtended_cff’)
process.load(’Configuration/StandardSequences/MagneticField_AutoFromDBCurrent_cff’)
process.load(’Configuration/StandardSequences/Reconstruction_cff’)
process.load(’Configuration/StandardSequences/FrontierConditions_GlobalTag_cff’)
process.load(’Configuration/EventContent/EventContent_cff’)

process.maxEvents = cms.untracked.PSet(
    input = cms.untracked.int32(100)
)

# Input source
process.source = cms.Source("PoolSource",
    # Specify your list of files here
    fileNames = cms.untracked.vstring(
        # include list of input files here
    )
)

# Output definition
process.output = cms.OutputModule("PoolOutputModule",
    splitLevel = cms.untracked.int32(0),
    fileName = cms.untracked.string(’output_file.root’),
    dataset = cms.untracked.PSet(
        dataTier = cms.untracked.string(’RECO’),
        filterName = cms.untracked.string(’’)
    )
)

# Other statements

# Specify updated Global Tags as necessary
process.GlobalTag.globaltag = ’GR09_R_35X_V4::All’  #
process.MessageLogger.cerr.FwkReport.reportEvery = 1

# Include UserDefinedBit0 in HcalSeverityLevelComputer

# Include the file ‘‘RemoveAddSevLevel.py’’ given in Appendix A
# in the same directory as this cfg file
import RemoveAddSevLevel

# Remove the HFLongShort bit from the Severity Level Computer
process.hcalRecAlgos=RemoveAddSevLevel.RemoveFlag(process.hcalRecAlgos,
                                                  flag="HFLongShort")
# Add UserDefinedBit0 with severity level 10
```

```
process.hcalRecAlgos=RemoveAddSevLevel.AddFlag(process.hcalRecAlgos,
                                               "UserDefinedBit0",
                                               10)
# Display revised computer
RemoveAddSevLevel.PrintLevels(process.hcalRecAlgos)

# Load the HF RecHit re-flagger
process.load("RecoLocalCalo.HcalRecAlgos.hcalrechitreflagger_cfi")
process.hfrecoReflagged = process.hcalrechitReflagger.clone()

# Use the re-flagged HF RecHits to make the CaloTowers
process.towerMaker.hfInput = cms.InputTag("hfrecoReflagged")
process.towerMakerWithHO.hfInput = cms.InputTag("hfrecoReflagged")

# Path and EndPath definitions
process.reflagging_step = cms.Path(process.hfrecoReflagged)
process.rereco_step = cms.Path(process.caloTowersRec*
                              (process.recoJets*
                              process.recoJetIds+
                              process.recoTrackJets)*
                              process.recoJetAssociations*
                              process.metreco) # re-reco jets and met
#process.rereco_step = cms.Path(process.towerMaker*
                                process.ak5CaloJets*
                                process.met) # a simpler use case
process.out_step = cms.EndPath(process.output)

# Schedule definition
process.schedule = cms.Schedule(process.reflagging_step,
                                process.rereco_step,
                                process.out_step)
```